SAIIE29 Proceedings, 24th – 26th of October 2018, Spier, Stellenbosch, South Africa © 2018 SAIIE

DEVELOPMENT OF A DEMONSTRATOR OF BIG DATA ANALYTICS

Rhett Desmond Butler[1*] & Prof. James Bekker[2]

[1]Department of Industrial Engineering
Stellenbosch University, South Africa
17647657@sun.ac.za

[2]Department of Industrial Engineering
Stellenbosch University, South Africa
jb2@sun.ac.za

ABSTRACT

Big Data Analytics is now not only being applied in the fields of science and business, but in healthcare and economic development, by organisations such as the United Nations. The research presented in this article provides a demonstration of developing a Big Data Analytics Demonstrator by integrating selected hardware and software. The components of such an analytics tool are presented, as well as the analysis of results of test data sets. Experience gained when setting up a proprietary data analytics suite is shared, and practical recommendations are made.

The goal of this demonstrator is to illustrate that a system could be built to provide meaningful insights into a given dataset, by making use of free-to-use software, commodity hardware and leveraging machine learning to mine the data for these insights.

[1] The author was enrolled for an MEng (Industrial) degree in the Department of Industrial Engineering, Stellenbosch University, South Africa
*Corresponding author

1. INTRODUCTION

Big Data Analytics is gaining widespread adoption due to the inherent properties that define Big Data, which provide the means to analyse large volumes of data in rapid fashion [1]. These properties are the four V's that define Big Data: *Volume, Variety, Veracity* and *Velocity*. These definitions were discussed in detail in [2] on which this article expands. The article by [2] investigated what Big Data Analytics (BDA) is, demonstrating the workings of the tools commonly associated with Big Data Analytics, namely, *MapReduce* and *Hadoop,* and provided an architecture which was used to develop a Big Data Analytics Demonstrator (BDAD).

This article expands on the research conducted by [2], by taking the idea and principles, and using it to aid the building and development of the BDAD outlined in this paper, using the open-source software Hadoop and Apache Spark. This paper therefore not only outlines what is Big Data, but also provides a demonstration of the technologies along with practical considerations when working with such a system. The Apache Spark software, which was developed in 2009 by [3] at UC Berkley, was built due to the need to analyse large volumes of data but with greater speed than what *MapReduce* offered. The method by which Apache Spark analyses data is the focus of the literature review in the following section, also providing reasons as to why this Analytics software was more applicable to the BDAD, and the ultimate goal for this project to better understand BDA.

The goal of the demonstrator developed in this project, was to illustrate how such a system could be built making use of available commodity hardware and open-source software to aid in decision-making. This is because documentation is scattered or lacking, which provides a concise picture of Big Data and all the components, associated with it. The demonstrator would do this by analysing given structured-datasets in a timely and accurate manner, using Big Data technologies and machine learning tools and techniques available. The results are then compared to those of standard analytics systems hardware and software which do not use Big Data technologies. Such technologies offer the means to analyse engineering-related data, from production quantities, worker productivity and the effect of different factory layouts on output. For this reason, an investigation and research was conducted into Big Data Analytics. BDA is a tool which aids the data analysis and decision-making of industries involved in the fourth industrial revolution.

The article is organised to first provide background around the Spark Analytics software used in this project in order to conduct the Big Data Analytics. Thereafter in section 3, the methods followed in order to develop the BDAD are outlined, discussing the practical challenges experienced during this phase of the project in order to develop the Big Data Storage and Analytics components required for such a system. Finally, in section 4, the demonstrator's capabilities are validated against a standard analytics system which does not make use of Big Data Analytics technologies. The validation is executed by comparing the time and accuracy taken to analyse ever larger datasets and making use of different machine learning (ML) tools and techniques as outlined in [2]. The validation also serves to demonstrate not only the Demonstrator's capabilities, but also the capabilities of BDA and why it should be implemented in more industries.

2. LITERATURE REVIEW

The literature review provides background to the workings of the Apache Spark Analytics software used in this project. The method by which Apache Spark analyses data is firstly discussed; then the different properties such as APIs (*Application Programming Interface)* and libraries of Apache Spark are outlined.

2.1 Apache Spark

To understand why Apache Spark (Spark) was developed, the *MapReduce* framework needs to be understood. As from [2], a *JobTracker* which is located on the master node, is used in *MapReduce* to manage the processing of a query (locate stored files, execute the jobs and store metadata of the processing). The *TaskTrackers*, which are located on the slave nodes, conduct the processing of a query that was submitted to the *JobTracker*. The query and the processing thereof is mapped to various partitions located on slave nodes which contain the source file. In order to manage this process, *MapReduce* makes use of *key-value* pairs (*k1,v1)*, where similar keys are joined, sorted and filtered in the *Map-phase*. The final step is to consolidate the results back from all the slave nodes to the master node via the shared *key-value* pairs, to where the *JobTracker* is located in the *Reduce-phase*, [2] and [4]. This process creates a lot of overhead, which increases the time per analysis [5] as each process is read and written to disk.

Because of this lengthy process, Spark was created. Spark makes use of in-memory processes in order to address this shortcoming of *MapReduce*. Spark uses a *Resilient* Distributed Dataset (RDD) for its processing. An RDD is a collection of data packets stored and distributed across a cluster in memory. This RDD, when consolidated, represents the data currently manipulated by an analyst. RDDs are immutable (cannot be altered, only a new copy can be created) which ensures fault-tolerance throughout a cluster by storing the metadata of the RDD copies. RDDs can be created in four ways [5], the first being loading a file from a shared file system such as HDFS (Hadoop Distributed File System) as discussed in [2], or loading an array generated previously within specifically the Scala programming language environment. Thirdly, by transforming an already existing RDD, or lastly, by persisting (caching) the RDD in memory or to disk (storing in a different file system).

The process to create and analyse data using an RDD within a cluster is given in Figure 1. Similar to the process used in *HDFS* and *MapReduce*, Spark has a *JobTracker* located on the Spark Driver. Located on Worker nodes or Slave Nodes are the *Executors*, these have *TaskTrackers* which execute processing of the RDD, sent to them from a *JobTracker*. As shown in the figure, there are many *TaskTrackers* conducting processing and sending the results back to the *JobTracker*, all of the processing occurring in memory [6] and [5]. For this reason, Spark is considered much faster than *MapReduce*; up to 100 times faster [3].

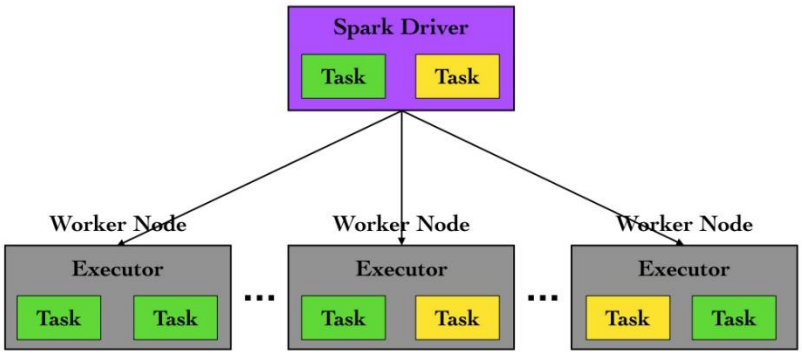

Figure 1: The process by which Spark submits jobs to a cluster of multiple nodes [3], making use of *JobTrackers* and *TaskTrackers* to execute and manage RDDs

Even though Spark is faster at conducting an analysis, it is not suited to all applications. As discussed in [7], *MapReduce* can be better suited when conducting linear large-scale processing that can be considered trivial, or when no immediate results are required, and therefore the rate of processing is of less importance.

Spark, at the time of writing, provides multiple APIs for different popular programming languages; these include R, Scala, Java and Python [3]. This allows pre-built programs to be easily adapted in the respective languages to make use of the Spark technology, increasing the user base. Spark also contains a pre-built library of ML tools (Supervised, Unsupervised ML *etc.*) and techniques (Linear Regression, k-means clustering *etc.*) for each of these languages. These additions negate the need for third-party ML applications and language packages that have to be added.

In the next section, the steps followed and practical challenges experienced to develop the BDAD cluster of computers are outlined. This includes connecting the hardware together over a network, and configuring the software, namely the Hadoop and Spark software used as the Big Data storage and analytics solutions.

3. BUILDING A BIG DATA ANALYTICS DEMONSTRATOR

There are multiple avenues available for an analyst to gain access to a Big Data analytics suite. These range from using pre-built solutions offered by companies such as Cloudera and Hortonworks [6] and [8]. These suites come with the relevant Big Data storage and analytics environments built-in and just have to be connected to an existing number of computing nodes. Another solution is to make use of cloud computing services offered by companies such as Amazon Web Services, which offer computing nodes on a wide variety of performance scales and prices (pay-as-you-use basis). These nodes are free to be configured however desired or have to be pre-built in a similar fashion. This makes it quick and easy to create a Big Data Analytics cluster in order to start with an analysis, without the need to configure the system and all its related components [9].

The goal of this project was to provide a demonstration of Big Data by making use of commodity hardware, and therefore a solution had to be configured which demonstrated how to make use of a combination of spare computing nodes and open-source software. For this reason, a pre-built system would not offer the means to learn how Big Data technologies work, and how they can be customised or expanded. The pre-built solutions are also better suited to companies and individuals looking to contract their analytics out and not develop in-house solutions. By building a demonstrator, the fundamental components could be demonstrated, making use of commodity hardware joined together to provide insights at a larger scale. By building this BDAD, a better understanding was formulated as to what Big Data Analytics entails. This included using technology to increase the rate at which an analysis of large datasets could be conducted, which is not possible using standard analytics software.

For this project, the BDAD made use of three computing nodes of which one was also the master node. The master node was responsible for process management and metadata storage and from which the queries were submitted to the cluster, the slave nodes being responsible for the computations of the query. The subsequent discussion is divided into hardware and software sections. In the hardware section, the computing nodes' hardware specifications are provided, as well as how the computing nodes were connected together on the network to form the BDAD cluster. Thereafter, in the software section, the operating system (OS) choice is discussed, outlining why the Big Data Analytics storage solution (HDFS) and analytics solution (Spark) were chosen, along with the process to configure these software applications together in a cluster. Finally, a list of practical challenges experienced during the development of the BDAD is provided, highlighting what to consider if such a system were to be replicated.

3.1 Hardware used in the Big Data Analytics Demonstrator

Figure 2 provides a visual representation of the configuration used by the BDAD. The term *NameNode* and *DataNode* will be used in this article interchangeably with master node and slave node.

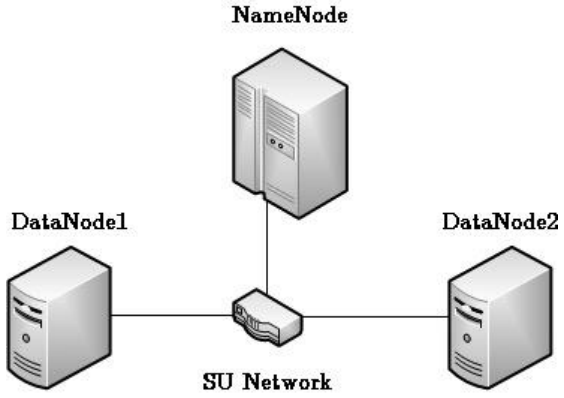

Figure 2: A representation of the three computing nodes used in the Big Data Analytics Demonstrator, connected together over the university network

The hardware used was chosen as it aligned with the scope of the project, to make use of commodity hardware. The specification for the three computing nodes is provided in Table 1. It is not suggested that use of this hardware configuration be made when developing a similar Big Data Analytics system in future, as due to technological advances over time, this hardware selection will be inefficient. This configuration was merely used at the time of this project as it aligned with the scope of this project. The information regarding the hardware specifications listed in Table 1 was from [10] and [11].

Table 1: The hardware specifications of the three computing nodes used in the Big Data Analytics Demonstrator

Hardware	Master Node (*NameNode*)	Slave Nodes (*DataNode 1 & 2*)
CPU	Intel 4th Generation Core i7-4770 with a clock speed of 3.4GHz	Intel Xeon 5500 series processors W3550 with a clock speed of 3.07GHz
RAM	Non-ECC dual channel 1600MHz DDR3 SDRAM, with 24 GB of memory	ECC 1333MHz DDR3 memory configured to 12 GB.

GPU	Integrated Intel HD Graphics 4600	Support for 2 PCI Express x16 Gen 2 graphics cards up to 150 W, NVIDIA Quadro 4000
Storage	One 500 GB disk drive (ST500DM0 02-1BD142 SCSI Disk Device) and one solid state drive of 256 GB (Samsung SSD SM841N 2.5 SCSI Disk Device)	One Disk Drive with 500 GB of storage (ST3500418AS ATA Device)
Networking	Integrated Intel I217LM Ethernet LAN 10/100/1000; supports optional PCIe 10/100/1000 network card	Integrated Broadcom 5761 Ethernet controller with Remote Wake UP and PXE support

The next step after acquiring the hardware was connecting the three nodes together over the Stellenbosch University network. The steps outlined were specific to the given OS used and may differ depending on the OS; however, the fundamentals remain the same. The next section outlines the steps followed as well as what to consider during configuration.

3.2 Connecting the computing nodes together over a network

The first step in the development of the cluster was to create unique identifiers for each node, namely *hostnames* that were used in place of the IP (Internet Protocol) address of a computer, to identify each on the network. Thereafter, creating a common user account, which has all the necessary permissions that is separate from the system accounts of the nodes, for consistency and good practice (should problems arise). Firstly, when configuring the cluster, a connection had to be established between the nodes, to allow for seamless data transfer, whilst being secure. For this project, the nodes' hostnames were:

- For the master node: *master*
- For the two slave nodes: *datanode1* and *datanode2*

Using these hostnames and IP addresses, the different nodes were located on the network. After establishing a linkage between the nodes, a secure connection, which would allow for data transfer without requiring passwords, was required. Using SSH (Secure Shell) technology, an encryption key was created and distributed between the nodes and the connection was tested. To test the connection, a *ssh accountname@hostname* (account name and hostnames created) command could be run to access another node securely and without a password, thereby creating the fundamental connections of the BDAD.

3.3 Software used for the Big Data Analytics Demonstrator

As mentioned in section 3.2, the OS used for the BDAD was Linux, specifically the CENTOS 7 distribution. The OS was chosen after experimenting with pre-built Big Data Analytics suite solutions by Cloudera and Hortonworks which both use Linux CENTOS OS, as well as after reading articles and blog posts by persons who have developed Big Data Analytics systems. Other OSs can also be used, as this article does not prescribe the exclusive use of Linux, but rather Linux was chosen as it was easily configurable with the Big Data technologies used for this project (Hadoop and Spark), and it is available as open-source. For the programming, the Python programming language was used, as it is easily interpreted and widely used by the scientific community, while being offered on Spark as an API, namely *PySpark*.

Following on from the literature review, the Spark analytics software was installed on the BDAD. The steps by which to configure the system were a combination of steps from [12], [13] and [14] providing the means to install the Spark software version 2.2.1, on the Big Data cluster. After this, the Spark environment was configured and the web dashboards to monitor spark jobs could be called by typing the following in the web browser:

- *master:8080* and,

In addition, when a query was executed:

- *master:4040*

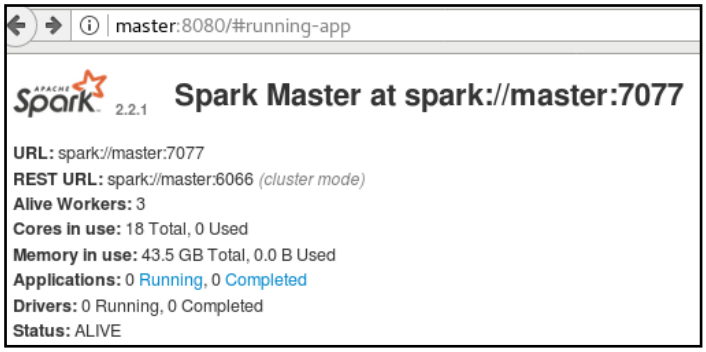

Figure 3: The Spark dashboard provided (master:8080), indicating the number of nodes (shown as workers) and cluster configuration along with the URL for calling the cluster

Using these dashboards, the system was inspected to see if the desired configuration was successful. For this project, a total of 43.5GB of RAM was configured (10.6GB per slave node and 22.3GB from the master node) per node and six of the eight CPU cores were configured for the Spark system per node (a total of 18 cores). This was validated by the output given in the master:8080 URL (*Uniform Resource Locator)* in Figure 3. To access the Spark cluster, the *spark://master7077* command is used at the start of a query.

The final software component used in this Big Data cluster was the HDFS storage solution from Hadoop. For this project, the Hadoop HDFS version 2.8.3 was used. The steps on how this distributed file storage system was configured were a combination of steps from [15], [16], [17] and [18]. The Hadoop program was then started, and then using the master:50070 command to view the Hadoop dashboard, the successful configuration was confirmed.

For this project, due to the limited number of storage and computational nodes, the master node was also used as a computing and storage node (not just to manage the processes typically associated with the master nodes). For this reason, the number of nodes indicated in Figure 4 (*Live Nodes*) which offer storage, was three and not two. The same applied for the analytics system, and why 18 cores were configured and not 12.

Configured Capacity:	796.99 GB
DFS Used:	1.42 GB (0.18%)
Non DFS Used:	24.62 GB
DFS Remaining:	770.95 GB (96.73%)
Block Pool Used:	1.42 GB (0.18%)
DataNodes usages% (Min/Median/Max/stdDev):	0.00% / 0.17% / 0.25% / 0.11%
Live Nodes	3 (Decommissioned: 0)

Figure 4: The summary provided by Hadoop, indicating the complete system storage configuration and the number of nodes used

Information provided by the Hadoop dashboards allow the analyst to monitor the files stored by the system, along with the number of data packets distributed across the system. As discussed previously in [2], the Hadoop system divides a file into data packets and stores and distributes these packets of information across the cluster to provide a fault-tolerant and scalable storage solution. The other dashboards to inspect the *DataNodes* were accessible by using the *hostname*:50075 (*hostname* of the node) command.

Other storage options that are open-source were also available for use, such as the MongoDB, Cassandra, Hive *etc*. However, it was decided to use the HDFS as it was a simple system requiring that data be stored in a distributed manner, and it was able to demonstrate the capabilities of distributed computing. The other storage solutions offer capabilities not required for this system, *e.g.* Cassandra offers high speed access suited to near-real-time analysis, or MongDB offers a NoSQL database, and they were therefore not chosen.

An advantage of using such Big Data technologies is that adding another computing node to the network requires that the current environment be stopped, and the new node's details and software files (Hadoop and Spark)

then simply have to be added and installed. The system then requires a restart after which the new node is then added to the cluster. This simple process allows more nodes to be added in parallel, allowing for scalability in storage and performance as desired. This means the computing power can be increased should the query complexity increase and the need for more computing power be required.

3.4 Practical challenges faced during the Demonstrator Development

The following is a list of the various practical challenges experienced during the development of the BDAD. The list provides a guide as to what should be considered during the development in order to reduce set-up time and improve the ease of development.

- During the period when the clusters of computers were being connected to a network, it was important to ensure that the global network IP address was selected, and not the nodes' internal IP (typically a 192.168 or 10.0 number), as the nodes were not then visible to each other and could not share data.
- The firewall had to be altered in order to allow the connection on the specific port through, as at first a connection was established, but information could not flow bi-directionally, only from the master node to the slave node and not in the opposite direction.
- Ensuring the SSH key was correctly stored and visible to create the connection was a common issue which prevented the system from allowing data transfer to occur.
- A practical challenge experienced during the Spark installation was configuring the different *DataNodes* to the cluster, as Spark would not connect these to the cluster due to incorrect read and write permissions (using Linux commands *chown* and *chmod* to give these permissions) given to the files where Spark locates the *DataNodes* and *NameNodes*.
- Similar to Spark, for Hadoop, the folders and their locations where the Big Data programs were located had to be given the correct permissions, as previously mentioned by using the *chmod* and *chown* commands. Hadoop during set-up would not otherwise be able to create the desired storage directories for the cluster.
- It must be ensured that all the Java packages were installed and be up to date, while the correct location of Java program files must be directed to Hadoop; it makes use of Java-related technologies to conduct its processes.
- A considerable amount of time was spent during the development phase to ensure correct configuration files (.xml files which provided settings, paths *etc.*) for the Hadoop environment were selected, as the example files found online are system specific and each file needed to be altered to suit the BDAD of this project.
- Before the Hadoop software was started, it was also important to use correct configuration files, specifying the number of times a file is replicated across the cluster. This has to be set to only the number of *DataNodes* configured, along with ensuring an appropriate block-size (size the file is broken up into) is used (*e.g.* if working with very large files opting for 128 Mb blocks instead of 64 Mb). By not doing this, duplicate files were stored on a node, which caused file corruption and then destroyed the HDFS storage system, preventing data from being read from the database.

In section 0, validation of the BDAD is presented. This was done by analysing ever-larger samples of a dataset stored in Hadoop, by applying different ML techniques under each tool. For each ML tool, the appropriate dataset was used during the analysis. The results of the Big Data system were then compared to that of a *standard* system, to compare the time taken to conduct an analysis and its accuracy. The validation serves to demonstrate the BDAD capabilities along with the benefits of using BDA.

The *standard* system, used to compare the Big Data Analytics Demonstrator to, was created by taking the master node, but not coupling it to a cluster. This *standard* system made use of an open-source ML library S*cikit-learn,* available in Python, in order to conduct the same analysis as that on Spark, but not in a parallel-distributed manner.

4. VALIDATING THE DEMONSTRATOR

The validation of the BDAD is divided into three sections; each taking the commonly used ML tools (Classification, Regression and Clustering*)* for each system under Supervised and Unsupervised ML. For each tool, a dataset containing data types that are appropriate for the tool was used, which are publicly available and found online. The datasets used in each tool were then divided into different sizes in an ascending order (1 000 rows, 500 000 rows *etc.*), taking random samples (datapoints) from the total dataset to fill each sample size with the specified

number of data points. This sample size, of e.g. 1000 rows of randomly selected datapoints from the complete dataset, was then divided randomly into training and testing sets. The training set consisted of 70% of the samples, and 30% for the testing set. These training and testing sets from the specific sample size was then used by the ML algorithm, and the results were recorded. This process was repeated 100 times for each sample size. The time and accuracy for each of the 100 samples was recorded and plotted, in order to identify and compare the results from the Big Data system with that of the Standard system (which uses the open-source Python ML library *Scikit-learn*), with regards to time and accuracy. Each sample

An example of this would be for Linear Regression. By taking dataset ***X***, which is the total dataset and randomly dividing the dataset into smaller datasets $\mathbf{x}_{(i;j)}$, where *i* is the sample *number i* ∈ {1, 2, …, n} and *n* = 100 samples, for dataset sample size *j* ∈ {1 000; 5 000; 50 000; 500 000; 1 000 000; 5 000 000; 10 000 000; 20 000 000}. For each of these 100 samples, the results were then plotted (time *vs*. accuracy), e.g. $\mathbf{x}_{(38;1\ 000\ 000)}$ would be the 38th sample of sample size 1 000 000 rows. For each sample, randomly selected datapoints were taken until the 1 000 000 rows for the sample were filled. This sample was then randomly divided into training (700 000) and testing sets (300 000) respectively, followed by analysis, and the time and accuracy for this 38th analysis being recorded.

To ensure for a controlled experiment on both systems, only the time and accuracy of each analysis was recorded, and not the time required performing the pre-processing. Along with this, only the necessary steps required during the ML analysis were added, such that the ML algorithm could perform the analysis. Any parameter to *e.g.* smoothen, reduce over-fitting or improve the accuracy of each individual analysis was not added, but simply running the data through the algorithm and recording its performance, to ensure comparability between the two systems ML algorithms.

The goal of this validation was to illustrate the benefits of using Big Data technology when working with larger datasets, the BDAD's ability to analyse larger datasets than those of the standard system; when it is required to obtain results in a timely and accurate manner, and also to validate the BDAD and demonstrate Big Data Analytics.

4.1 Results from the classification analysis

The different Classification algorithms used to test the BDAD were chosen as they are commonly used and require different data types, which would allow the system capabilities to be tested and compared to the standard system. The datasets used were chosen to ensure the data types of the variables/features were labelled and contained categorical features which would allow for classification. An example where such classification algorithms could be used in engineering is to classify faulty or non-faulty products after production, given a set of features that describe what constitutes a faulty item.

4.1.1 Decision Trees

Due to its popularity, the first classification algorithm namely Decision Trees (DT) was considered. The dataset used to analyse both systems was a freely available NYC taxi dataset. The dataset had 28 million rows of taxi trips made for the first three months of 2017. The dataset contained 17 features, of which the *passenger_count* (number of people in the taxi for the specific trip), *trip_distance* (in miles), *Ratecode_ID* (what rating the ride was, dependent on trip surcharges *etc.*), *payment_type* (form of payment made) were used to predict the cost of the trip to the customer. The feature used to predict the cost was *fare_amount* (cost per trip in USD), the trip cost was divided into 15 cost ranges (from [0-2], [2-4],…,[400-450] *etc.*), in order for the DT ML algorithm to classify which cost range a customer could fall into given the input features. The dataset after pre-processing had 21 million usable rows from which to make predictions, therefore the samples were divided into sample sizes 1 000, 5 000, 50 000, 500 0000, 1000 000, 5000 000, 10 000 000 and 20 000 000. This was to get a broad idea of both systems' performance capabilities with different sample sizes. One hundred randomly selected samples were chosen for each sample size and a prediction was made whether a person would fall into one of the respective ranges, given the set of input parameters.

The results from the BDAD and standard systems are as follows. Figure 5 shows the mean results from the analysis on the standard system. As can be seen, as the sample sizes increased the mean accuracy of predictions increased. The BDAD was able to conduct the analysis in a reasonable time, but as can be noted in Figure 5, the time taken to analyse the sample size of 20 million was considerably longer at around 580 seconds per sample analysis.

Upon further investigation of the standard deviation shown in **Table 2**, the general trend indicates a small standard deviation in accuracy, therefore predictions did not differ by much from the mean for the BDAD. The *Standard* system was able to provide results that were generally lower in accuracy and with a larger standard deviation, shown in **Figure 6** and **Table 3** respectively. The standard system was, however, able to complete the analysis in less time than the BDAD. In addition, the standard system was not able to analyse the final sample size of 20 million, outputting a memory error.

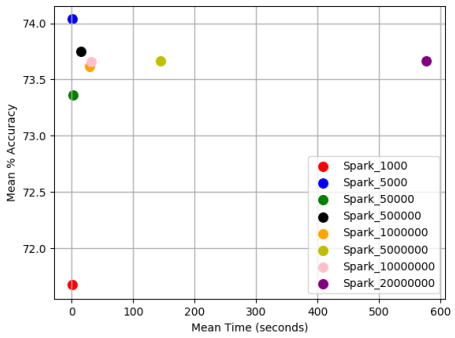

Figure 5: The mean results for the BDAD using the DT. As noted, the larger the sample size, the more accurate the prediction accuracy

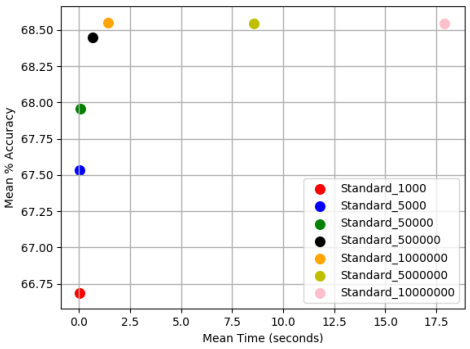

Figure 6: The mean results from all sample sizes on the Standard System. As noted, the larger the sample size, the more accurate the prediction accuracy

Table 2: Table of the standard deviations of the prediction accuracies for the BDAD for the DT

Sample Size	Standard Deviation
1000	0.0292
5000	0.0114
50000	0.0040
500000	0.0012
1000000	0.0034
5000000	0.0006
10000000	0.0006
20000000	0.0008

Table 3: Table of the standard deviation of the prediction accuracy for the DT on the Standard System

Sample Size	Standard Deviation
1000	2.8762
5000	1.4801
50000	0.7871
500000	0.3884
1000000	0.1420
5000000	0.0416
10000000	0.0257

4.1.2 Naïve Bayes

The same dataset that was used in demonstrating the *Decision Trees* ML algorithm was used for the Naïve Bayes ML algorithm. As with the NYC taxi dataset, the samples were divided randomly as before, with the same sample sizes for the 100 samples per sample size.

The results from the analysis on the BDAD and Standard system are shown in Figure 7 and Figure 8. From the figures, the mean accuracies for the predicted values of *fare_amount* decreased for the BDAD, and for certain sample sizes for the standard system. Spark predicted accuracy was lower than that on larger datasets, but critically, was able to analyse the larger datasets (the standard system was unable to analyse the 20 million-row dataset), compared with the standard system. The standard deviations were as before lower on the BDAD in Table 4 compared to the standard system shown in **Table 5**, which indicates that even though the accuracy was lower; the values remained close to the mean value. The reason for the lower prediction accuracy could be due to the dataset not being optimal for the algorithm on Spark; Naïve Bayes is typically applied to text classification.

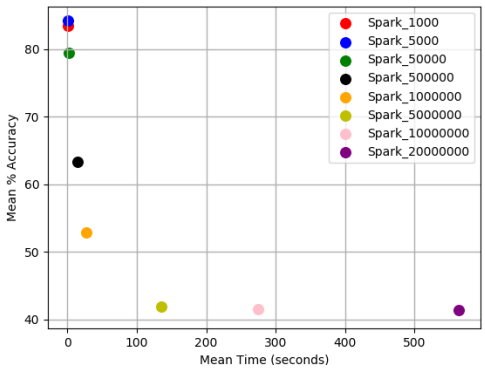

Figure 7: The mean accuracy results from the Naive Bayes analysis on the BDAD

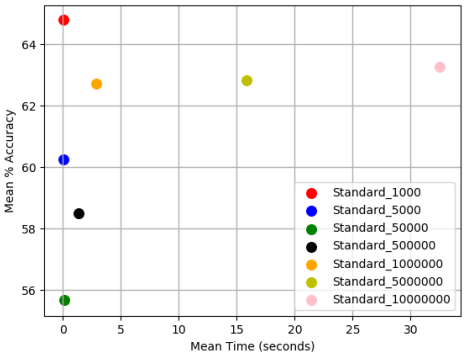

Figure 8: The mean accuracy results from the Naive Bayes analysis on the Standard system

Table 4: Table of the standard deviations of the prediction accuracies for the BDAD

Sample Sizes	Standard Deviation
1000	0.0260
5000	0.0186
50000	0.0758
500000	0.1954
1000000	0.1802
5000000	0.0009
10000000	0.0007
20000000	0.0008

Table 5: Table of the standard deviation of the prediction accuracy for the Standard System

Sample Size	Standard Deviation
1000	4.8724
5000	4.9056
50000	5.0318
500000	7.9217
1000000	4.8256
5000000	2.8657
10000000	2.3236

4.1.3 Logistic Regression

The dataset used in order to test and demonstrate the BDAD to the standard system was a freely available dataset, 23 features of which were used to indicate whether a woman would be likely to have a child which is born live, or *still/have an abortion*. The dataset had 7 259 473 rows. The dataset was a collection of results from questionnaires from a study conducted in different states in India. The dataset was modified so that if it was determined the woman had a birth which was *live*, a value of one was assigned, and zero for *still/abortion*. Logistic regression requires binary classification given a set of multiple features; therefore, the dataset used in *Decision Trees* was not applicable. The sample sizes were divided 1 000, 5 000, 50 000, 500 000, 1 000 000, 2 000 000, 4 000 000, and 6 000 000 accordingly, and as before, 100 samples per sample size were taken and analysed by the BDAD and standard system.

The results from the analysis on the BDAD and standard system show high accuracies were achieved as shown in **Figure 9** for the BDAD, with the standard system performing better with smaller dataset sizes as expected (the BDAD is designed for large datasets) shown in **Figure 10**. The standard system could only analyse sample sizes of up to 2 million rows before encountering limitations due to the hardware. The standard deviation also shows better performance with smaller datasets in **Table 6** compared with **Table 7** . From the results, it was shown that the standard system provides accurate results, but it could not analyse larger datasets, which was the purpose of the study, to demonstrate the limitations of non-Big Data software compared to the BDAD developed by the author.

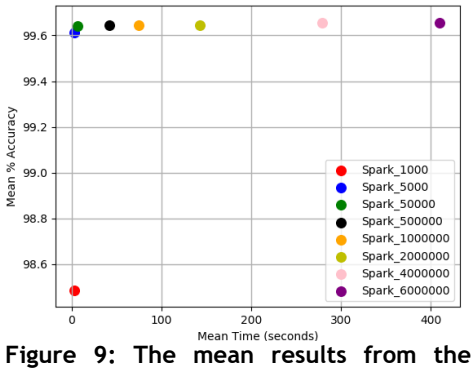

Figure 9: The mean results from the Logistic Regression Analysis on the BDAD

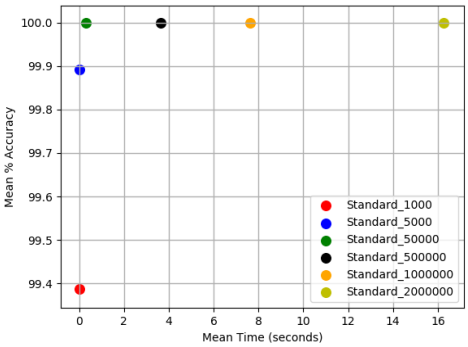

Figure 10: The mean results from the Logistic Regression Analysis on the standard system

Table 6: Standard deviations of the prediction accuracy of Logistic Regression on the BDAD

Sample Size	Standard Deviation
1000	0.0123
5000	0.0012
50000	0.0004
500000	0.0001
1000000	0.0001
2000000	0.0001
4000000	0.0000
6000000	0.0000

Table 7: Standard deviations of the prediction accuracy of Logistic Regression on the Standard System

Sample Size	Standard Deviation
1000	0.0034
5000	0.0006
50000	0.0000
500000	0.0000
1000000	0.0000
2000000	0.0000

4.1.4 Support Vector Machines (SVM)

The same dataset, sample, and sample size divisions used in the analysis of *Logistic Regression* for the BDAD and standard system, were used in the *SVM* analysis. SVM and Logistic Regression are typically being applied in text classification and image recognition, and therefore the datasets used to demonstrate the BDAD were chosen due to the size to demonstrate use cases of BDA. Therefore, these datasets are not optimal, but due to the difficulty to locate large enough datasets, these datasets were chosen.

The results in

Figure 12 show that for small sample sizes, the standard system outperforms the BDAD, but as sample sizes increased, the standard system was unable to perform a SVM analysis and only up to 500 000 rows could be analysed. This demonstrates the ability of the BDAD to make predictions on large datasets as all sample sizes could be analysed shown in Figure 11 with good accuracy compared to the standard system. The standard deviation values of the standard system in **Table 9** indicate the better performance at smaller sample sizes compared to that of the BDAD in **Table 8**.

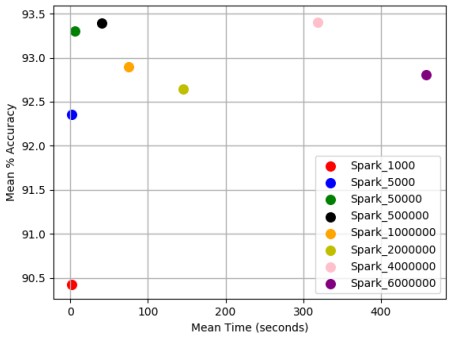

Figure 11: The mean results from the SVM Analysis on the BDAD

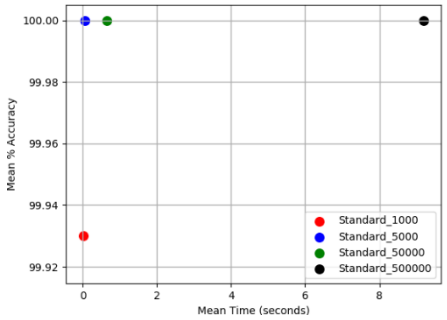

Figure 12: The mean results from the SVM Analysis on the standard system

Table 8: Standard deviations of the prediction accuracy of SVM on the BDAD

Sample Size	Standard Deviation
1000	0.0208
5000	0.0115
50000	0.0203
500000	0.0189
1000000	0.0140
2000000	0.0099
4000000	0.0189
6000000	0.0000

Table 9: Standard deviations of the prediction accuracy of SVM on the Standard System

Sample Size	Standard Deviation
1000	0.0024
5000	0
50000	0
500000	0

4.2 Results from the Regression analysis

To demonstrate the BDAD capabilities on regression analysis, a Linear Regression analysis was conducted with both systems. The datasets used for this were chosen such that the data contained data types that were labelled and numerical. A use case example in engineering could be to use regression analysis to predict production output over time producing a number of products on a production line.

4.2.1 Linear Regression

The analysis of the BDAD capabilities in conducting a Linear Regression analysis was tested using the same NYC taxi dataset as for the Decision Tree analysis. The same sample sizes and number of samples per sample size were used. For this analysis, however, the relationships between the *trip_time*, *trip_distance* and *fare_amount* were analysed and a Linear Regression analysis applied to predict how much a person could expect to pay for a taxi trip given the time and distance they expected to travel from the current position. This is because NYC taxis take into account time and distance when calculating the cost for a fare (time used when taxi is still, distance for when the taxi is moving).

The results from the analysis showed higher predicted values for the standard system in Figure 14 than the BDAD in Figure 13, with the standard system having slightly higher standard deviation values in Table 11, compared to the BDAD in Table 10. This means the predicted values vary more around the mean for the standard system. However, it can be noted that the standard system was not able to analyse the data for a sample size of 20 million, whereas the BDAD was able to provide results with small standard deviations. This analysis shows that the BDAD is able to analyse a larger dataset than the standard system, whilst providing accurate results in a timely manner.

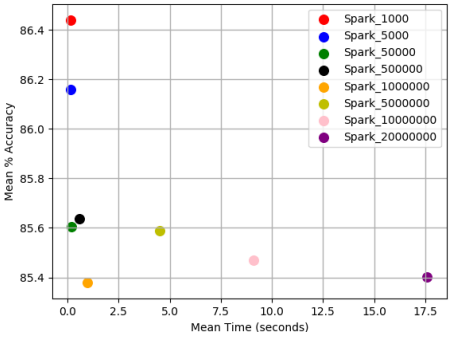

Figure 13: The mean results from the Linear Regression Analysis on the BDAD

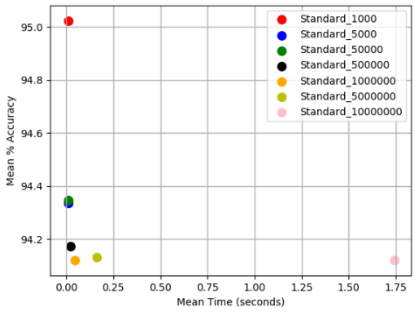

Figure 14: The mean results from the Linear Regression Analysis on the standard system

Table 10: The mean results from the Linear Regression Analysis on the standard system

Sample Size	Standard Deviation
1000	0.0800
5000	0.0358
50000	0.0168
500000	0.0055
1000000	0.0059
5000000	0.0018
10000000	0.0023
20000000	0.0011

Table 11: Standard deviations of the prediction accuracy of Linear Regression on the Standard System

Sample Size	Standard Deviation
1000	0.0472
5000	0.0320
50000	0.0166
500000	0.0055
1000000	0.0053
5000000	0.0023
10000000	0.0019

4.3 Results from the Cluster analysis

The k-means clustering ML algorithm was used to demonstrate the capabilities of the BDAD. This algorithm requires data to be unlabelled and numerical. Because the data is unlabelled, no predictions are made, but the algorithm investigates the similarities between the data points, grouping common data points together in clusters given their similarities with one another. An example would be to perform a cluster analysis of retail customers given a set of features, grouping similar customers together.

4.3.1 k-means

The dataset used to analyse the clustering capabilities of the BDAD was a freely available dataset of radiation readings taken from across the world totalling 80 million readings. The reason for using this dataset was due to the volume of data it contained, which could be clustered, namely the geographical positions of all the readings taken. Due to the unsupervised nature of clustering, the analysis of the clusters was done by making use of elbow plots, which required visual inspection in order to determine the best number of clusters after the clustering algorithm had been run.

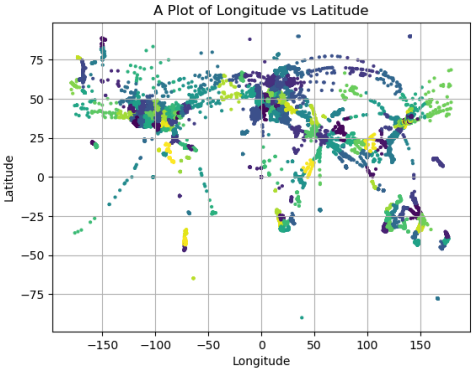

Figure 15: 195 Clusters formed after k-means on the BDAD. Only displaying a sample of the 80 million points

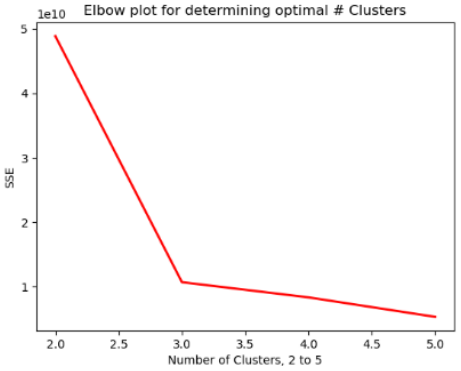

Figure 16: The elbow plot showing the SSE, indicating best number of clusters to be three on the BDAD. The plot is for illustration purposes, so as to identify the elbow in the plot

The results from performing k-means on the BDAD are shown in Figure 15 and Figure 16. These are the preliminary results indicating all 195 clusters, which represent the current 195 countries worldwide. The reason for choosing this number is to conduct an investigation into the radiation levels per country (taking into consideration also that some readings were taken by planes and ships, as shown in the figure, which clearly should not belong to a 'countries cluster' and is a contextual error of the clusters). When performing k-means in order to determine the best number of clusters used by the *elbow* method, these methods use the SSE (Sum Square Error) to determine the error in the distance between the datapoints and cluster centroid. The resulting *elbow plot* given in Figure 16 shows the optimal number of clusters to be three. In the context of the data provided, given that the analysis is correct and three clusters were optimal, the three clusters were deemed not to provide information of any further use. Figure 15 shows the radiation clustered roughly according to country, providing an analyst with information around radiation levels per country (making the results more usable).

The standard system was unable to perform *k-means* clustering on the dataset and was therefore unable to provide any meaningful insights. This example also demonstrated the superiority of the BDAD over the standard system in analysing larger volumes of data.

This dataset showed that the analysis undergone needs to be context specific, and that the acquired results need to be questioned for their applicability to the problem or question being posed.

5. FUTURE WORK

For the BDAD, proposed future work is to deploy more ML algorithms on the BDAD cluster in order to test the system capabilities. Another option could be to create a distributed non-relational database using technologies such as MongoDB. Using this, a datastore can be created, which offers the means to store data, not only in a distributed manner but also to access relevant data directly. This is because currently the database used for this project stores and distributes a file containing the relational information, which has to first be pre-processed to access the relevant fields. Another option is to test and use fast access databases such as Cassandra, in order to test the system using near real-time data and conduct near real-time data analysis, instead of batch analysis as was done for this project.

6. CONCLUSION

The BDAD developed for this project and as outlined in this article, demonstrates the use case for Big Data technologies; in order to conduct Big Data Analytics, typically unavailable or misunderstood by many. This article serves to illustrate to industrial engineers how such a BDA system could be developed using commodity hardware and software. The various analyses demonstrated the capabilities of the system compared to a *standard* system, which makes use of non-Big Data technologies in order to analyse data, using machine learning.

The results indicate the ability of the BDAD to provide accurate predictions with larger datasets than the standard system, thus demonstrating the capability of the BDAD and Big Data Analytics.

Such a system is of benefit to engineering, as more data is generated than previously and requires deeper analysis in less time. Deploying BDA and machine learning to aid analysis in fields where industrial engineering practices are required can benefit engineering as a whole. The information could then aid the decision-making processes of industrial engineers. This project and the BDAD is seen as a tool which can be used by systems designed with the fourth industrial revolution in mind, by providing BDA to discover deeper insights to aid decision making as well as automate and manage the processes of these new systems.

REFERENCES

[1] **P. C. Zikopoulos, D. DeRoos, K. Parasuraman, T. Deutsch, D. Corrigan, and J. Giles**, *Harness the Power of Big Data*. 2012.

[2] **R. D. Butler and J. Bekker**, "Development of a Big Data Analysis Demonstrator," in *SAIIE 28th Annual Conference*, 2017, pp. 1–13.

[3] **A. Spark**, "Apache Spark," *Apache Spark*, 2018. [Online]. Available: http://spark.apache.org/. [Accessed: 01-Feb-2018].

[4] T. White, *Hadoop: The Definitive Guide*, 3rd ed. Sebastopol, CA: O'Reilly, 2012.

[5] **M. Zaharia**, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark : Cluster Computing with Working Sets," *HotCloud'10 Proc. 2nd USENIX Conf. Hot Top. cloud Comput.*, p. 10, 2010.

[6] **Cloudera**, *Spark Guide*, 1st ed. 1001 Page Mill Road, Bldg 3 Palo Alto, CA 94304: Cloudera, 2017.

[7] **A. Bekker**, "Spark vs. Hadoop MapReduce: Which big data framework to choose," *ScienceSoft*, 2017. [Online]. Available: https://www.scnsoft.com/blog/spark-vs-hadoop-mapreduce. [Accessed: 01-Mar-2018].

[8] **Hortonworks**, "Hortonworks Data Platform," 2016.

[9] Amazon, "Amazon Web Services," *Amazon*, 2018. [Online]. Available: https://aws.amazon.com/. [Accessed: 10-Jun-2018].

[10] **Dell**, "Dell Optiplex 9020 Specifications." Dell Computers, p. 3, 2018.

[11] Dell, "Dell Precision T3500 Specifications." Dell Computers, p. 2, 2018.

[12] **DataDotz**, "Spark Master/Slave installation in Multi Node," *Chennai Hadoop User Group*, 2018. [Online]. Available: http://chennaihug.org/knowledgebase/spark-master-and-slaves-multi-node-installation/. [Accessed: 02-Mar-2018].

[13] **S. Mitra**, "How to Setup Hadoop Multi Node Cluster - Step By Step," *DWBI*, 2016. [Online]. Available: https://dwbi.org/etl/bigdata/201-install-spark-in-hadoop-cluster. [Accessed: 02-Mar-2018].

[14] **DeZyre**, "Step-by-Step Apache Spark Installation Tutorial," *DeZyre*, 2018. [Online]. Available: https://www.dezyre.com/apache-spark-tutorial/apache-spark-installation-tutorial. [Accessed: 02-Mar-2018].

[15] **Y. Darji**, "Easily Setup Multi-Node Hadoop Cluster in YARN Mode on CentOS/Ubuntu," *Medium*, 2017. [Online]. Available: https://medium.com/@yogeshdarji99/installing-multinode-hadoop-cluster-in-yarn-mode-86741c3df45b. [Accessed: 27-Feb-2018].

[16] **Fibrevillage**, "No Title," *Fibrevillage*, 2018. [Online]. Available: http://fibrevillage.com/storage/617-hadoop-2-7-cluster-installation-and-configuration-on-rhel7-centos7. [Accessed: 27-Feb-2018].

[17] **DWBI**, "How to Setup Hadoop Multi Node Cluster - Step By Step," *DWBI*, 2016. [Online]. Available: https://dwbi.org/etl/bigdata/183-setup-hadoop-cluster. [Accessed: 27-Feb-2018].

[18] **K. Rahul**, "How to Set Up Hadoop Multi-Node Cluster on CentOS 7/6," *tecadmin*, 2013. [Online]. Available: https://tecadmin.net/set-up-hadoop-multi-node-cluster-on-centos-redhat/. [Accessed: 27-Feb-2018].